



MASTER OF TECHNOLOGY

PRACTICAL LANGUAGE PROCESSING (PLP)

MovieBot

Movie Sentiment Analysis and Recommender

Team Members

Student Name	Student ID
Lim Jun Ming	A0231523U
Gopan Ravikumar Girija	A0231541U
Tadhg Kennedy	A0231552N

Contents

1	Executive Summary	3
2	Introduction	3
3	System Design	4
4	Task-Oriented Chatbot Architecture	5
4.1	Natural Language Understanding (NLU)	5
4.2	Natural Language Generation (NLG)	9
4.3	Dialogue Management (DM)	14
4.4	Chatbot Design Limitations	16
5	Aspect-Based Sentiment Analysis on Movie Reviews	17
5.1	Background	17
5.2	Dataset	17
5.3	Aspect-Based Sentiment Analysis Pipeline	18
5.4	Evaluation Results	22
5.5	Deployment	22
6	NLP Recommender system	23
6.1	Dataset	23
6.2	Experimentation	24
6.3	Evaluation	26
6.4	Deployment	26
6.5	Limitations	27
7	User Interface & Integration	28
8	Conclusion	29
9	Future Improvements	29
10	References	30
11	Appendix A: Handcrafted Inform Intents Dataset	32
12	Appendix B: Handcrafted Control Code Dataset for Text Generation	33
13	Appendix C: BERT for Joint Intent and Slot Classification Fine-Tuning Setup	34
14	Appendix D: T5-small model Pre-Training and Fine-Tuning setup for text generation	34
15	Appendix E: Reddit App Creation Steps	35
16	Appendix F: Installation Instructions	36

1 Executive Summary

Modern movies are seeing a division between critics and fans on a more frequent basis. With the advent of social media movie goers are more often seeking the opinions of the masses over those of critics. Additionally, the sheer volume of movies which are becoming available through streaming services can be overwhelming for consumers when it comes to selecting a movie to watch. However, there isn't any one stop platform available which can help consumers address these problems. We propose a dialogue system which can analyze the sentiment of audiences on social media and provide movie recommendations to users based on their preferences. The dialogue system follows a task-oriented chatbot architecture to achieve the objectives set out above. We explore various approaches and methods when designing the dialogue system, such as Natural Language Understanding/Natural Language Generation architectures, aspect-based sentiment analysis, word embedding based search. Finally, we are able to implement a functioning prototype which is able to converse with a user and perform the tasks outlined above.

2 Introduction

With today's technology and the increase in the user interaction online, the importance of reviews by professional critics has dwindled as more and more people seek the broader opinions of everyday people when selecting their entertainment media. The difficulty with this lies with the disparate sources of reviews, from formal websites like IMDb to casual user reviews on Twitter or Reddit. Pulling this information together and more importantly gaining valuable insight is a difficult task for the production companies and users alike. With the age of streaming services hitting full swing the volume of movies and TV shows becoming available yearly, monthly, or even weekly is hard to keep up with. This incredible variety can be a blessing and curse. Users can often find shows that really appeal to them, but it can be difficult to even know where to start. For example, user can often spend 30 minutes just scrolling through the options because they can't decide what they want to watch.

This project seeks to address these problems using a dialogue system:

1. Conducting real-time aspect-based sentiment analysis from multiple social media platforms to provide insight into the opinions of the general audience about a desired media.
2. Providing movie recommendations based on the users self-described preferences using Natural Language Processing methodology.

As a Minimum Viable Product (MVP) we seek to design a functioning chatbot system, by exploring various approaches and methods to address the problem statement. For a task-oriented chatbot we look at the classic architecture which contains the four key components: NLU, Dialogue States Tracking, Dialogue Policy and NLG. We investigate the use of aspect-based sentiment analysis approach to extract fine-grain insights into the audience opinion from Reddit and to provide summarization. In addition, we explore the use of word embeddings and transformers-based models to develop a recommender model using the self-described preferences of the user as input.

The chatbot system will be deployed in the form of a flask web application and is capable of understanding users requests accurately and interact with the user by generating fluent and natural responses. This facilitates users to interact with the chatbot to query for the public's opinion of movies as well as for the users to get recommendations based on users' preference.

3 System Design

The figure 1 below illustrates the overall system design of MovieBot.

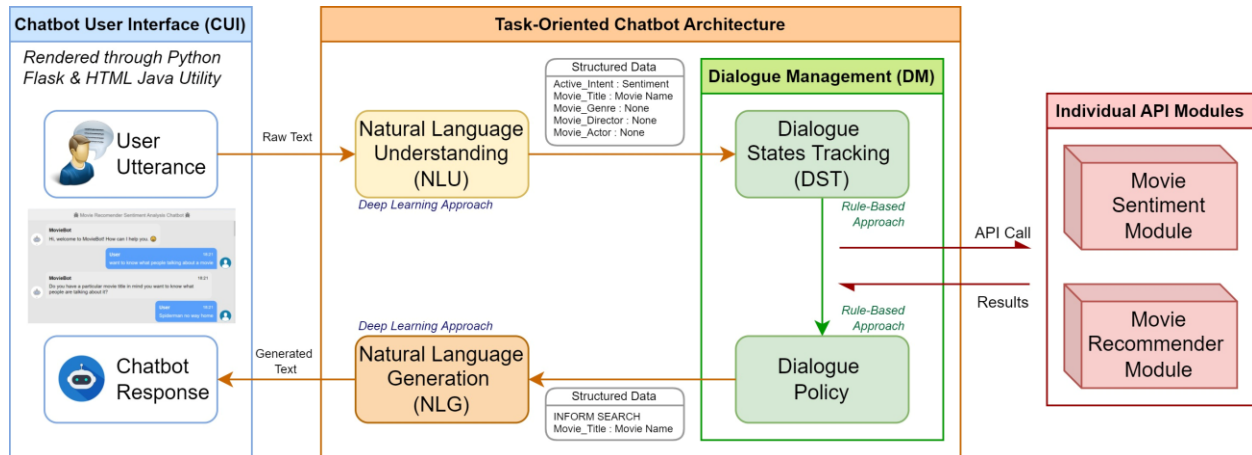


Figure 1. MovieBot System Design

The system modules are separated into 3 groups, the Chatbot User Interface (CUI), the Task-Oriented Chatbot Architecture and the Individual API Modules.

The CUI serve as a user interface where the user interacts with the system in the form of text dialogues. The user interface is a simple a webapp rendered using the Python Flask and HTML Java utilities. User raw text input are captured in the CUI and then is sent to the chatbot modules for processing.

Every time a user utterance in raw text is being sent to the chatbot modules, it will first be processed under the Natural Language Understanding (NLU) module to extract the user intents and any relevant slot information. The NLU module will then output the intent and slots extracted in the form of structured data and feed them into the Dialogue Management (DM) module. Within the DM module, there is a rule-based Dialogue States Tracking (DST) function that updates the current states that represents the dialogue history leading up to current conversation round. Depending on the current states and whether the necessary slot values are filled sufficiently, the system will trigger the API call to the respective modules to run and retrieve the results of movie sentiment analysis or movie recommendations. Given the updated states and/or API call results, a rule-based Dialogue Policy module will then decide on the next system action. The selected system action in the form of structured data will then be fed into the Natural Language Generation (NLG) module to generate system response in human natural language.

Finally, the generated text from the NLG module will be sent back into the CUI and will be printed as system or chatbot response to interact with the user.

Overall, the full cycle of the chatbot modules is run recursively whenever there is new user utterance being captured from the CUI. The main objective of respective modules within the system is to achieve the specific tasks, movie sentiment analysis and movie recommendations in our case; by understand the user utterance while at the same time keeping track of dialogue history and to generate fluent and natural text responses to interact with the user.

4 Task-Oriented Chatbot Architecture

4.1 Natural Language Understanding (NLU)

Background and Related Works

Natural Language Understanding (NLU) has been one of the classic tasks in the Natural Language Processing (NLP) domain. The focus of NLU is for machine to interpret or understand the meaning and context of a given text or speech through syntactic and semantic analysis. The NLU is also a growing research field in recent years due to abundance of its downstream NLP applications like summarization, question answering, named-entity recognition (NER), machine translation as well as utilization in virtual assistant or dialogue related systems. For the scope of this project, we are specifically interested in the application of NLU in a task-oriented chatbot system. There are 2 critical tasks that the NLU would need to perform for the chatbot system:

1. Intent Classification – To identify the main intention of the user utterances
2. Slots Filling – To capture the relevant slots or elements in the user utterance

Traditionally, the intent classification and slot filling is seen as two different problems in NLU and different approaches are explored to process the two tasks independently. The intent classification is often treated as a text classification problem. The mainstream approach[1-2] include machine learning models like Naïve Bayes, Support Vector Machine (SVM) which may requires feature engineering and deep learning models like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM, Bi-LSTM) as well as transformer family models like BERT and RoBERTa. On the other hand, the slot filling is view as a sequence labelling problem where token level classification is done based on BIO tags or labels. Machine learning approach for slots filling task include conditional random fields (CRF), Hidden Markov Models (HMM) and deep learning approach like CNN, RNN, LSTM as well as transformer-based models like BERT[2].

In recent years, the joint model approach for intent classification and slots filling has become a more popular architecture[3] as it has achieved state-of-the-art performance in NLU tasks especially with the rise of personal assistant or chatbot applications in our daily life. The joint model approach is proven to perform better here because there usually exist relationships between the intents and slots that can be learnt efficiently in a joint model approach. This is more apparent in the chatbot area where the intents and slots are usually closely related to a specific domain where the chatbot operates in.

However, there are still many challenges that plague many of the approaches and methods discussed above. Below are some challenges highlighted in the context of NLU tasks in chatbot system:

1. Lack of domain specific dataset
2. Ambiguity and variations in user expressions
3. NLU tasks become increasingly more challenging when chatbot system cover more functions and domains

Model Approach and Architecture

For the scope of our MVP, we will be adopting the joint model approach for intent classification and slot filling. The objective here is not to create a better state-of-the-art model but to fine-tune a model that works well enough for our chatbot application which involves the domain in Movie Sentiment Analysis and Movie Recommendations. Specifically, we refer to the approach discussed in the paper titled BERT for Joint Intent Classification and Slot Filling[3]. The figure 2 below illustrates the model architecture used for developing the NLU module for the MVP.

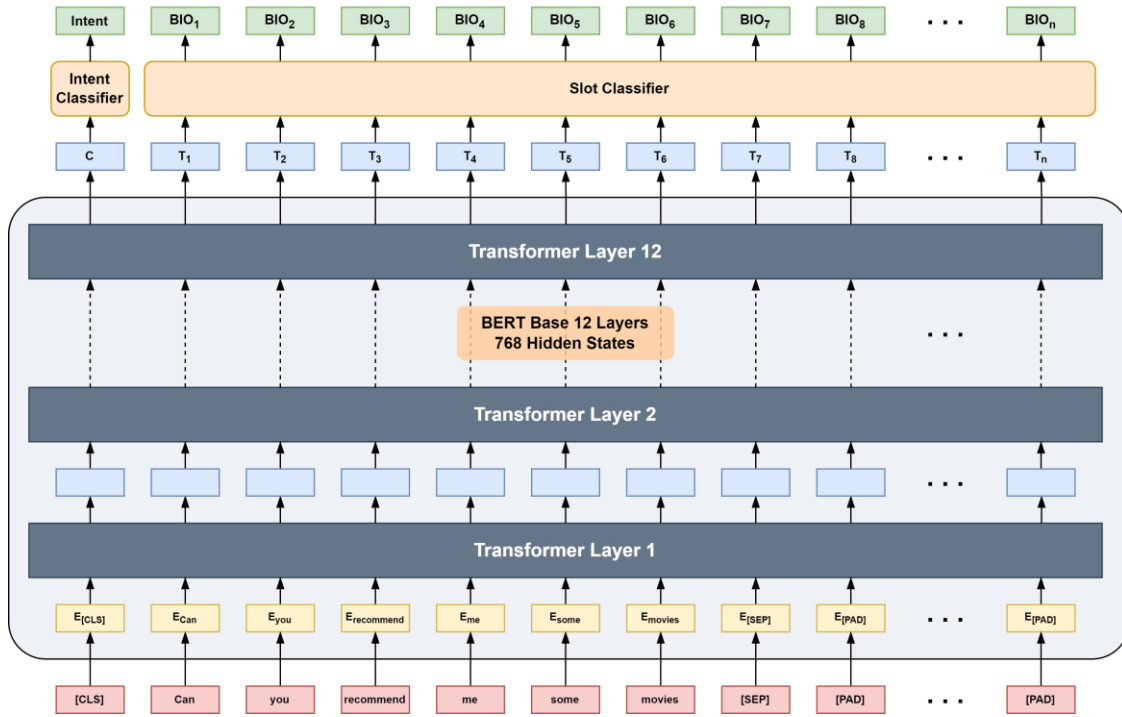


Figure 2. Joint Intent and Slot Classification using BERT Base

The base model used here is the BERT Base, Uncased model with 12 encoder layers, 12 attention heads and 110M parameters[4]. The hidden states size is 768 dimensions. An intent classifier is attached to the class label or the pooled output (1 x 768 dimension) while a slot classifier is attached to the sequence or token output (token length x 768 dimension) of the BERT model. The intent classifier is essentially a linear layer which predicts the intent of the input sentence or utterance. On the other hand, the slot classifier is a linear layer which predicts the BIO labels for each token in the input sentence or utterance. The number of intent classes and number of BIO label types depends on the dataset that is being used to fine tune the model. The loss function for the model is the sum of the intent classification loss and the slot classification loss. The intent classification loss used is the categorical cross entropy loss of intent classes. The slot classification loss here refers to the categorical cross entropy loss of BIO labels for active tokens which exclude all special tokens like “[CLS]”, “[SEP]” and “[PAD]”.

$$Loss = Intent_{CELoss} + ActiveSlot_{CELoss}$$

where $ActiveSlot_{CELoss}$ is only for tokens $\neq [CLS], [SEP], [PAD]$

Dataset

Like many research work in the NLU area, domain specific datasets on intent classification and slot filling related to movie sentiment and movie recommendations are not readily available for use anywhere. Most of the open-sourced datasets for intent and slot filling like SNIPS[5] and CLINC150[6] only covers a few domains like restaurants, hotels, flights, etc. which are not useful in our case. However, these open-sourced datasets also contain user utterances with common intents such as “confirm”, “inform”, “negate”, “thank you” which can be useful for most chatbot systems.

To fine-tune the BERT model for joint intent and slot classifications, we have handcrafted dataset for domain specific samples on movie sentiment and movie recommendations. On the other hand, we have also selected some samples of common intents from the CLINC150 dataset. Together, they form a dataset of 790 samples with 8 unique intent classes and 9 BIO labels. Each data sample consists of 1 user utterance raw text, an intent label and BIO labels for each token of user utterance. The tokenizer used here is the pretrained BERT Base, Uncased tokenizer. Due to handcrafting custom dataset can be very costly, we have only created inform intents dataset where there is at most a single entry for each of the slot types (title, genre, director, and actor) for the scope of our MVP. Refer to **Appendix A** for more details on handcrafted dataset generation. The tables below show the details of final dataset used to fine-tune the model.

Dataset by Intent Types:

Intent Types	Samples	Source	Description
INFORM_INTENT SENTIMENT	80	Handcrafted	User target objective is related to movie sentiment analysis tasks
INFORM_INTENT RECOMMENDER	250	Handcrafted	User target objective is related to movie recommendation tasks
INFORM	250	Handcrafted	User informing relevant slot values
CONFIRM	40	CLINC150	Confirm, agree, or accept
NEGATE	40	CLINC150	Decline or disagree
THANK_YOU	40	CLINC150	Thank you intent
CLOSING	40	CLINC150	User wants to end the conversation
OUT-OF-SCOPE (OOS)	50	CLINC150	Intent could not be identified or intent that is out of scope
Total	790		

Dataset by BIO Labels:

BIO Labels	Token Count	Description
B-title	80	Beginning token for movie title
I-title	149	Intermediate token for movie title
B-genre	240	Beginning token for movie genre
I-genre	58	Intermediate token for movie genre
B-director	240	Beginning token for movie director name
I-director	554	Intermediate token for movie director name
B-actor	240	Beginning token for movie actor name
I-actor	552	Intermediate token for movie actor name
O	4,641	Outside Token (not related to any required slots)
Total	6,754	

Model Training Evaluations

The pre-trained BERT model used is loaded from the transformer library by HuggingFace[7] and the fine-tuning is performed using the Pytorch library in python.

Refer to the **Appendix C** for more details on the setup of the model fine-tuning process and parameters. The figure 3 below shows the train-validation loss and accuracy for fine-tuning the BERT model for joint intent and slot classification.

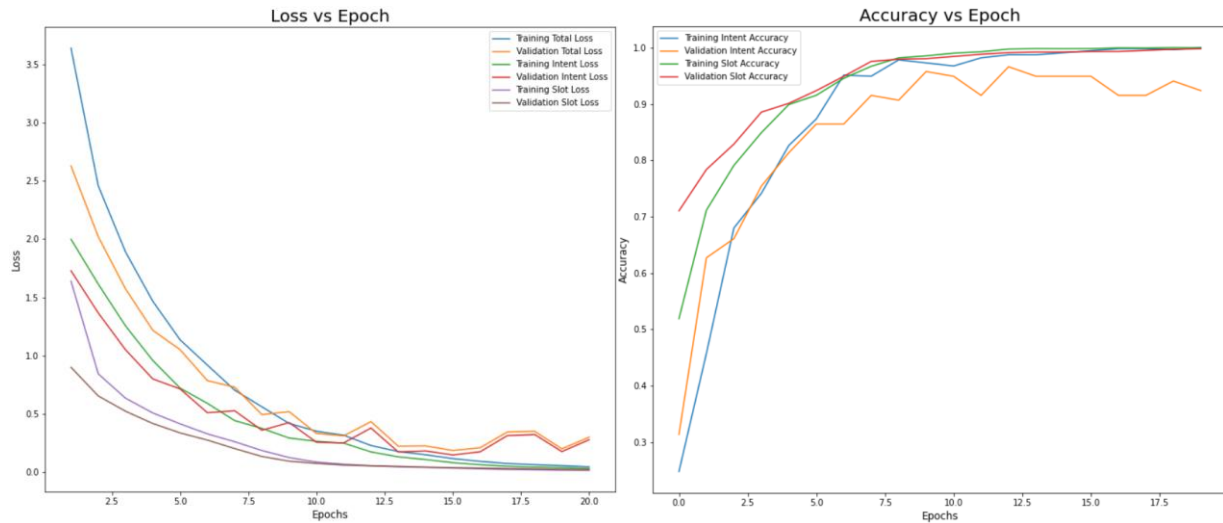


Figure 3. Fine-Tuning Loss and Accuracy

Based on the loss chart, there is no overfitting detected. We can also see that the validation accuracy is above 90% after the 5th epoch. The model callback used here is based on the lowest total loss.

To evaluate the performance of the fine-tuned model on both intent classification and slot filling task, we examine the model results based on the unseen test dataset. The precision, recall and f1 score of the intents and slot labels are important metrics to look at when evaluating the model performance. Below tables shows the evaluation results based on the test dataset.

Intent Classifications:

Intent Types	Precision	Recall	F1-score
INFORM_INTENT SENTIMENT	1.00	1.00	1.00
INFORM_INTENT RECOMMENDER	1.00	1.00	1.00
INFORM	1.00	1.00	1.00
CONFIRM	0.83	0.83	0.83
NEGATE	1.00	0.83	0.91
THANK_YOU	1.00	1.00	1.00
CLOSING	0.86	1.00	0.92
OOS	1.00	1.00	1.00

Test Dataset Accuracy = 0.98

Slot Classification:

BIO Labels	Precision	Recall	F1-score
B-title	0.89	0.89	0.89
I-title	1.00	0.95	0.97
B-genre	1.00	0.97	0.99
I-genre	1.00	1.00	1.00
B-director	1.00	1.00	1.00
I-director	1.00	1.00	1.00
B-actor	1.00	1.00	1.00
I-actor	1.00	1.00	1.00
O	1.00	1.00	1.00

Test Dataset Accuracy = 0.99

Based on the evaluation results, the fine-tuned model seems to perform decently with accuracy above 98% for both intent and slot classification. We can also see that model is able to predict many of the intent classes and BIO labels correctly when evaluated on the unseen test dataset.

Hence, the fine-tuned model is deployed and used for NLU module in the MovieBot system. For inference, both intent classes and BIO labels are saved in their corresponding labels to index dictionaries. The model will take in the raw user utterance as input and returns the predicted intent index and the BIO index for each of the word piece tokens. The predicted intent label can then be lookup from the labels to index dictionary. Similarly, to extract the slot values, we utilize the offset mapping output of the BERT tokenizer as well as the labels to index dictionary for BIO labels to obtain the exact words or phrases for each slot types detected.

4.2 Natural Language Generation (NLG)

Background and Related Works

Natural Language Generation (NLG) is another important subfield under the broad domain of Natural Language Processing (NLP). The focus of NLG is to generate fluent and natural text based on some data input which is the opposite what NLU does. Under the hood of NLG, there are many different types of text generation tasks such as text or topic summarization, article generation, and chatbot response generation. Specifically, we are interested in controllable text generation for our use case here. Even within the topic of controllable text generation, there are plenty of research directions which involves different kinds of controllable aspect in text generation such as attributed-based generations where the tone or sentiment are being controlled, dialogue generation where the persona and emotion are being controlled as well as storytelling where we would want to control the topic and the flow of the story.[8]

In our case of a task-oriented chatbot, we are interested in the data-to-text generation where the model takes in some form of structured data as input to generate fluent and natural text. The controllable aspect in the case of a chatbot response would be the slot values or exact keywords which we want them to appear in the generated text. As these slot values or keywords usually comes in the form of structured data, the common large pretrained language model usually will not perform here as the pretraining task was not conducted based on structured data inputs.

Traditionally, the approach to chatbot response generations usually depends on domain experts to design and handcraft response templates. The chatbot system will then fill in the slot values in the lexicalization post-processing step to create a chatbot response. The responses created using these template-based methods are not always fluent and can be very repetitive if the selection of template is small. We see in more advanced approaches such as the Context-aware LSTM (CA-LSTM)[9] where the slot values are injected into the decoder layers of the LSTM model in an attempt to include the slot values in the decoded text. In another research, Semantically Conditioned LSTM (SC-LSTM)[10], additional control vector and control cell are added into the LSTM architecture in an attempt to generate a controllable utterance which can then be lexicalized by filling in the slot values. In more recent research, we see that transformer-based models[11-14] are being used instead for data-to-text generation.

Overall, the data-to-text generation is still a developing research field in the NLG area. Currently, there is no open sourced pretrained model for the task. Similar to NLU, there are still many challenges that are faced by many of the text generation methods discussed above such as:

1. Lack of annotated domain specific system utterance dataset
2. It is hard to explain how would deep learning models be able to capture and understand the controllable aspect correctly in text generation

Model Approach and Architecture

For the scope of the project, we focus on a few research papers which have similar approach in developing a feasible data-to-text generation model. The approach is a 2 steps process:

1. Using a language model pretrained on plain text, further pretrained it on large corpus of multi-domain dialogue acts to system utterance dataset for the model to obtain the ability of system utterance generation based on encoded structured data input.
2. Further fine-tune the pretrained model on small dialogue acts to system utterance dataset which is specific to the chatbot application domain.

We have adopted the approach that is similar to the paper titled Text-to-Text Pre-Training for Data-to-Text Tasks[13] which is currently one of the state-of-the-art approaches in data-to-text generation for task-oriented chatbots[15]. We chose the Text-To-Text Transfer Transformer (T5)[16] as the language model here. We then pre-train the T5 model using a large corpus of annotated system utterance dataset. Using the pre-trained T5 model, we further fine-tune it using custom handcrafted dataset that is related to the domain of movie sentiment analysis and movie recommendations. The T5 model used here is the smallest model size (T5-small, 60M parameters) of the T5 model family.

The figure 4 below illustrates the model approach for the NLG module:

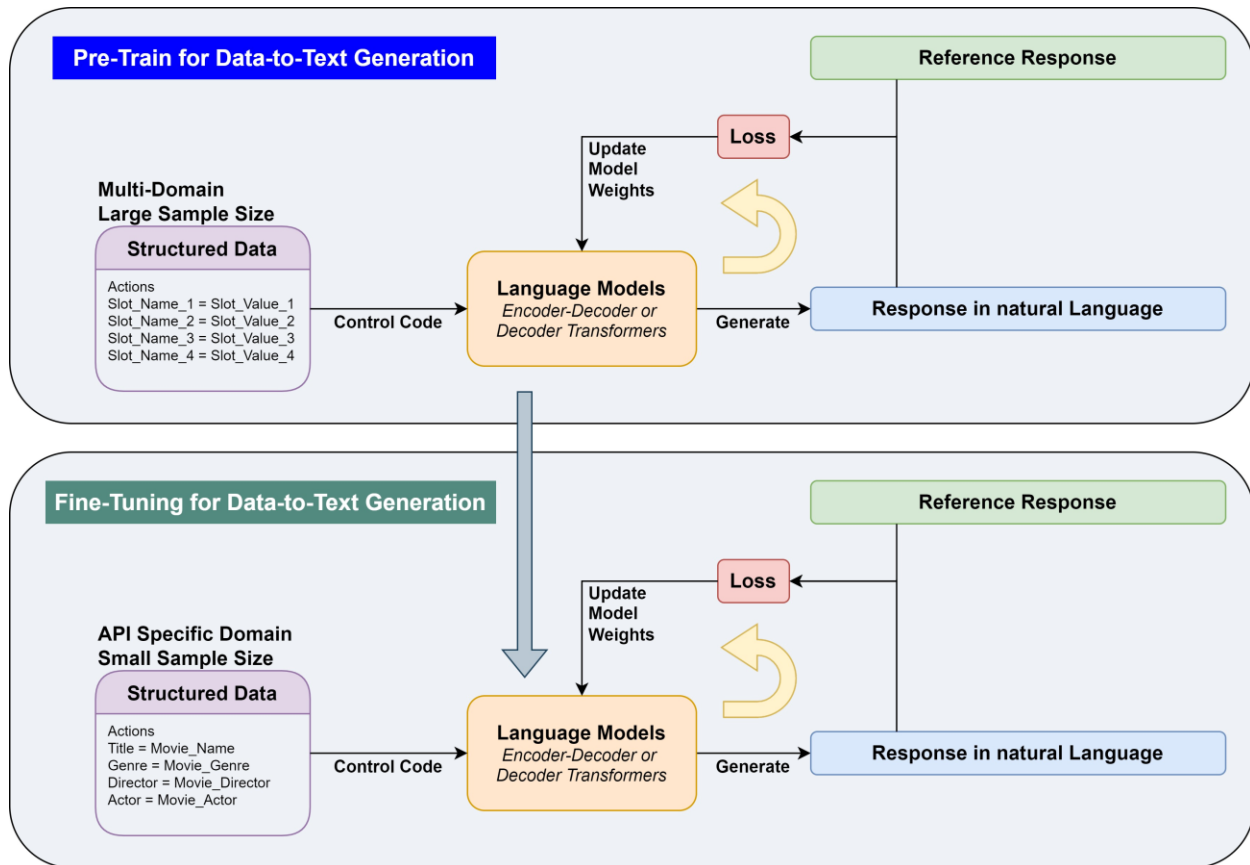


Figure 4. Pre-Training and Fine-Tuning for controllable text generation model

For both the pre-training and fine-tuning steps, the T5 model only takes structured data in the form of control code as input to generate text responses. The control code in our case is a pseudo string sentence that includes information like API domain, system action, slot names and slot values which are the key aspects we want to control in the text generation. Below are some examples of how system actions in the form of structured data are converted into control codes as inputs for the model.

Structured Data (System Actions) -> Control Code

Structured Data	Control Code
Domain – Flights Actions – REQUEST Slot_Name 1 – departure_data Slot_Value 1 – ? Slot_Name 2 – origin_city Slot_Value 2 – ?	Flights REQUEST (departure_data = ? ; origin_city = ?)
Domain – MOVIE_SENTIMENT Actions – NOTIFY_FAILURE Slot_Name 1 – title Slot_Value 1 – Back to the Future	MOVIE_SENTIMENT NOTIFY_FAILURE (title = Back to the Future)

Datasets

To pre-train the T5 model for data-to-text generation, we use the large open-sourced dialogue dataset called Schema-Guided Dialogue (SGD) Dataset[17]. It consists of over 20k annotated multi-domain, task-oriented conversations between human and a chatbot or a virtual assistant system. The dataset covers 20 domains such as Banks, Flights, Hotels, Travel, Car Rentals, etc. On system utterances, it has 10 distinct system actions and over 100 types of slots defined across the 20 domains. We are able to extract about 200k system utterances from the SGD dataset which can be used to pre-train the T5 model for data-to-text generation as it fits the requirement of large corpus of multi-domain annotated dialogue dataset as discussed above. Below is a summary of the extracted system utterances from the SGD dataset which are used for the pre-training step:

Schema-Guided Dialogue Dataset	
Domains	20 domains (Banks, Flights, Hotels, Travel, Car Rentals, Restaurant, Alarms, etc)
Actions	10 actions (confirm, goodbye, inform, notify failure, notify success, etc)
Slots	122 slot types across all domains
Samples	231,642 system utterances

On the other hand, for the further fine-tuning step, we have the same issue with NLU module where there is no open-source dialogue dataset that is related to movie sentiment analysis and movie recommendations. Therefore, we have handcrafted the system utterance dataset that can be used for further fine-tuning the T5 model for data-to-text generation. The table below is a summary of the handcrafted dataset which consists of 6 types of system actions and the relevant slots for the movie sentiment or movie recommendations. In our case, the possible slots are the specific slot values that we want them to appear the generated text. Refer to **Appendix B** for more details on handcrafted dataset generation.

Handcrafted Dataset (Movie Sentiment & Movie Recommendation Specific)		
Actions	Possible Slots	Samples
REQUEST	title, genre, director, actor	200
NOTIFY_SEARCH	title, genre, director, actor	210
NOTIFY_SUCCESS	title, genre, director, actor	210
NOTIFY_FAILURE	title, genre, director, actor	210
REQ_MORE	-	100
GOODBYE	-	100
Total		1,030

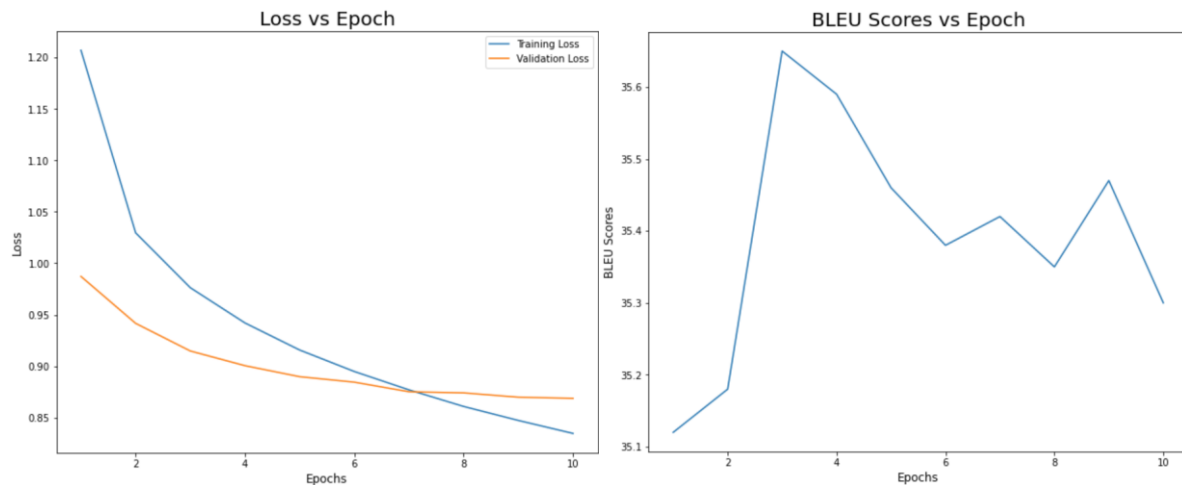
Model Training Evaluations

Similar to NLU model training, the pre-training and further fine-tuning of the T5-small model is conducted through the transformer library by HuggingFace[7] using the Pytorch library in python. Refer to the **Appendix D** for more details on the setup of the model pre-training and fine-tuning process.

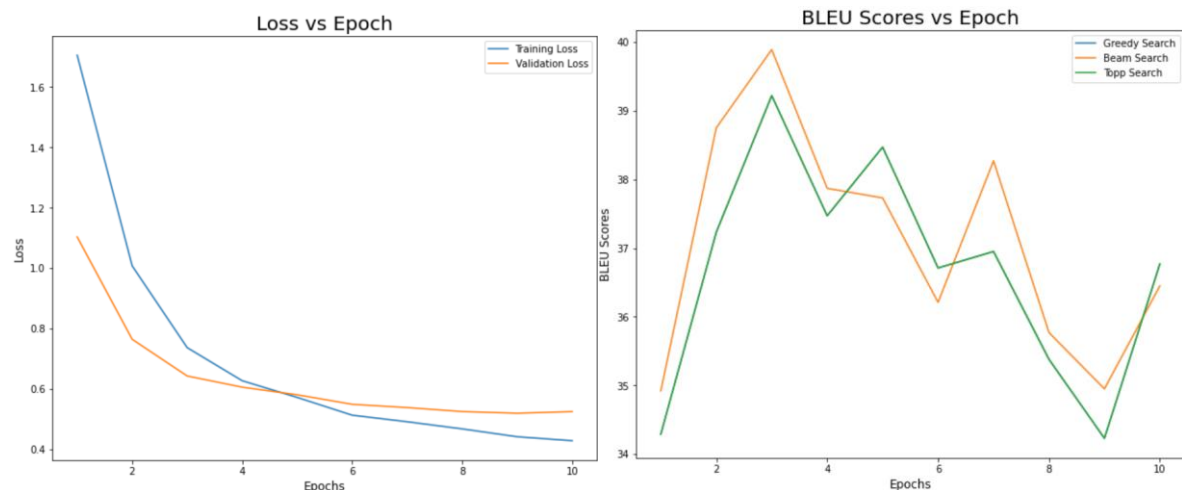
To evaluate the performance of text generation model, the Bilingual Evaluation Understudy (BLEU) metric is used here. The BLEU score is a metric that measures how close are the generated text is to the reference text in terms of word choice and word order. Higher score would mean the generated text is

more similar or identical to the reference text and this would ensure that the slot values are being mentioned in the generated text. For the pre-training step, we look at the BLEU score of the validation dataset based on greedy search decoding strategy. For the fine-tuning step, we look at the BLEU score of the validation dataset based on three decoding strategies: greedy search, beam search and top-p (nucleus) sampling. The figures below shows the train-validation loss and BLEU score of the pre-training and fine-tuning process.

Pre-Training T5-small with control code derived from system utterances of the SGD Dataset:



Further Fine-Tuning T5-small with control code derived from handcrafted system utterance dataset:



Below is a quick summary of the best validation BLEU score based on different decoding strategies:

Model	Dataset	Best Validation BLEU Score	
Reference Model (Original Paper) [13]	MultiWoz 2.1	Greedy Search	34.6
Pre-Training T5-small	SGD Dataset	Greedy Search	35.7
Further Fine-Tuning T5-small	Handcrafted Dataset	Greedy Search	39.2
		Beam Search	39.9
		Top-p Sampling	39.2

Based on the validation set BLEU score results, we can see that the performance of the pre-training of T5-small model on SGD dataset at 35.7 is close to the performance of the reference model in the original paper pre-trained on MultiWoz2.1 dataset at 34.6. Although the MultiWoz2.1[18] is a smaller dataset (71k system utterances) as compared to SGD dataset (over 200k system utterances), we consider the T5-small model pretrained on SGD dataset is sufficient for the scope of our MVP as the BLEU score results of both dataset approach are similar.

By performing further fine-tuning on the T5-small model pretrained on SGD dataset, we achieved slight improvement in the BLEU score at above 39 for all three decoding strategies tested. By performing some manual human evaluation, we can see that the model is capable of generating fluent and natural system utterance within the movie sentiment and movie recommendation domain where slot values are mentioned accurately. Below are some examples of text generation using simulated control code inputs.

Simulated Control Code (Input)	Decoding	System Utterance Generated by Model
MOVIE_RECOMMEND NOTIFY_SEARCHING (genre = <u>romance</u>)	Greedy Search	Looking for <u>romance</u> movies
	Beam Search	Looking for some <u>romance</u> movies
	Top-p Sampling	Looking for <u>romance</u> movies
MOVIE_RECOMMEND NOTIFY_SUCCESS (genre = <u>crime</u> ; director = <u>Someone Name</u> ; actor = <u>Another Name</u>)	Greedy Search	Below are some <u>crime</u> movies directed by <u>Someone Name</u> and acted by <u>Another Name</u>
	Beam Search	Below are some <u>crime</u> movies directed by <u>Someone Name</u> and acted by <u>Another Name</u>
	Top-p Sampling	Below are some <u>crime</u> movies directed by <u>Someone Name</u> and acted by <u>Another Name</u>

Finally, we have selected the model based on beam search decoding strategy for the NLG module as it has the highest validation BLEU score at 39.9. For deployment, the model will always take in the control code as defined above as input and generate system utterance which can then be send back to the frontend user interface as chatbot response.

4.3 Dialogue Management (DM)

With a working NLU and NLG model, the DM module will then play the important role of keeping track of the dialogue history while at the same time decide the next action or response the system should produce for the dialogue or conversation. As the number of user interactions required to achieve the objective of movie sentiment analysis or movie recommendations is relatively simple in our MVP, we have decided to adopt the rule-based approach for the Dialogue Management (DM) module.

Within the DM module, there are 2 key functions that will keep the conversation running:

1. Dialogue State Tracking (DST) – update and track the states of conversation
2. Dialogue Policy – decide the next system actions

Every time a user utterance is captured and processed through the NLU module, the DST will update the conversation states based on the extracted intents and slots. With the updated states, the dialogue policy function will then decide the next action for the chatbot system. The selected system actions in the form of predefined control code with corresponding slot values filled in will then be sent to the NLG to generate a chatbot response.

Combining the rule-based function of DST and dialogue policy, the state planning is designed based on the finite state machine approach. This means at any point of the conversation, there will be a state that is being assigned to represent the history of the conversation. The state in our case is a collection of data indicators which includes the current API domain that the conversation is on, the corresponding slot values in the form of data dictionary as well as indicators that shows whether there is sufficient information to run API calls or indicator for which the API call results return is successful or failure.

Figure 5 below illustrates the conversation planning that is designed based on the finite state machine approach with the built-in rule-based DST and dialogue policy functions.

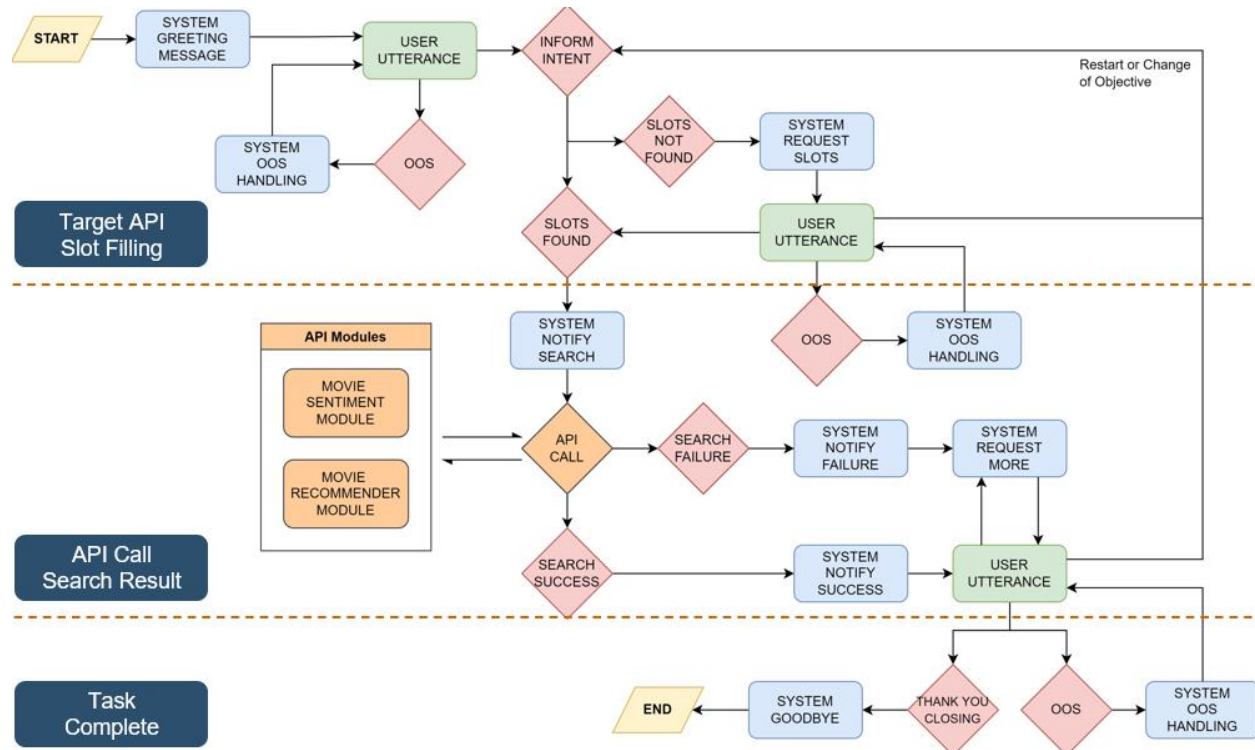


Figure 5. Conversation Planning with Rule-Based DST and Dialogue Policy

The conversation planning starts with a system greeting message and consists of 3 steps process to achieve the target objective of movie sentiment and movie recommendations:

1. Getting the target API (Movie Sentiment or Movie Recommendations) and filling in the necessary slots
2. Run the API feeding in the necessary slot values and retrieve the search results
3. Task complete once the results are presented to the user

We can see that at every possible user utterance turn, there is an out-of-scope (OOS) intent handling dialogue policy built in. This helps to keep the conversation to run within the predefined domain. Every time a state update (red cells) is incurred, a system action (blue cells) will be initiated by the dialogue policy in the conversation planning. All system action defined above except the system greeting message and OOS handling will be sent to the NLG to generate a chatbot response. Both the system greeting message and OOS handling have predefined text response which do not need to be processed with NLG module. In addition, the DST and dialogue policy rules are also designed such that at any point of user

utterance round, the user can also either reset or change the slot values of the current active API domain or to change the active API all together. The DST will reset the slot values or the active API domain correspondingly. The table below shows the description of the system actions that are built-in with dialogue policy function within the DM module. Note that the list of system actions that the dialogue policy function can choose from also heavily depends on the handcrafted dataset being used to fine-tune the T5 model in the NLG module. The selected system action should be one of the system actions which the NLG has been trained or fine-tuned on to ensure accurate chatbot response generation.

System Actions	Description
REQUEST	System utterance to request for required slot values corresponding to the objective API
NOTIFY SEARCH	System utterance to notify user that system is performing the API call or run
NOTIFY SUCCESS	System utterance to notify user that system has successfully perform the API call
NOTIFY FAILURE	System utterance to notify user that system has failed or did not find any relevant results from API call
REQUEST MORE	System utterance to request for user's next target objective
GOODBYE	System utterance to greet user goodbye

In summary, together with all modules in the chatbot architecture, the chatbot is able to understand the user's request through the NLU module, updates the conversation states and decide the system action in the DM module, and finally, to convert the selected system action into fluent and natural text response in the NLG module. This will be a full cycle of how the chatbot system interacts with its user on each round of conversation turns.

4.4 Chatbot Design Limitations

Below are some limitations identified for the chatbot design discussed above:

1. Dynamicity of conversation is very limited due to both NLU, and NLG models are trained or fine-tuned using custom dataset which has limited types of intent, slots and system actions.
2. The NLU module may not perform well all the time due to the ambiguity and variations of user expression used in user utterance.
3. The rule-based Dialogue Management module will become more complex as more functions and API domain is added to the chatbot system.

5 Aspect-Based Sentiment Analysis on Movie Reviews

5.1 Background

With the advancement of new online platforms, people started to express their views and opinions about a film through social media platforms like Twitter, Reddit, etc. Social media platforms like Reddit have many subreddit platforms that organize official discussions about a newly released movie and many people express their opinion through these official discussion threads. Our Aspect Based Sentiment Analysis feature helps the user to understand the general sentiments of the public by providing a fine-grained sentiment analysis report on each category, prepared by analyzing the opinion of people under the Reddit's official discussion thread for that movie.

5.2 Dataset

In this project, we present an upgraded version of Movie20 and the moviesLarge dataset presented in [19]. The original Movie20 dataset consists of 780 sentence, aspect term, and polarity pairs annotated manually by human annotators. We manually extracted the phrase describing each aspect term in the sentence and introduce that as an additional column of information in the upgraded version of the Movie20 dataset. Table 1 shows a few example instances from the upgraded Movie20 dataset. The upgraded version of the Movie20 dataset was used to test the performance of our Aspect Term Detection and Aspect Polarity Detection modules.

The moviesLarge dataset presented in [19] consists of 8732 sentence, aspect term and polarity pairs annotated automatically using a deep learning model which was trained on other benchmark datasets. Since it was not manually annotated this dataset had many wrong annotations for aspect term and polarity pairs. To improve the quality of the moviesLarge dataset we manually verified each sentence, aspect term, and polarity pair and re-annotated the data. While annotating the polarity we followed the high Precision and low Recall strategy i.e., positive polarity was assigned to highly positive phrases, the negative polarity was assigned to highly negative terms, and the rest are annotated as neutral. In addition to that, we also extracted phrases describing each aspect in the sentence and provided an additional column of information in the upgraded version of the moviesLarge dataset. Table 2 shows a few example instances from the upgraded moviesLarge dataset. The upgraded version of the moviesLarge dataset was used to train our Aspect Polarity Detection module.

In addition to that, we also used IMDb sentence polarity data [24] that consists of 1000 sentence, polarity pairs annotated manually by human annotators. We used this dataset along with the updated moviesLarge dataset to train our Aspect Polarity Detection module.

Sentence	Aspect Term	Phrase	Polarity
However, I think the plot is kind of lame.	[plot]	[kind of lame]	[Negative]
Great Acting dreadful editing.	[Acting, editing]	[Great Acting, dreadful editing]	[Positive, Negative]
A fantastic animated movie like-able characters fun to watch :)	[movie, characters]	[fantastic animated movie, like-able characters fun to watch]	[Positive, Positive]

Table 1. Example instances from Movie20 dataset.

Sentence	Aspect Term	Phrase	Polarity
Other than that, a predictable and boring action extravaganza.	[action]	[predictable and boring]	[Negative]
Excellent tone, well acted, and has some great twists.	[tone, acted, twists]	[Excellent tone, well acted, has some great twists]	[Positive, Positive, Positive]
This is a film by Steven Spielberg.	[Steven Spielberg]	This is a film by Steven Spielberg.	[Neutral]

Table 2. Example instances from moviesLarge dataset.

5.3 Aspect-Based Sentiment Analysis Pipeline

Figure 6 illustrates the overall pipeline of our Aspect-Based Sentiment Analysis module. Aspect-Based Sentiment analysis is a fundamental task in mining opinions and sentiment analysis. This task requires detecting the aspects of the target entities mentioned (Aspect Term Extraction) and detecting the sentiment attached, i.e., the polarity of the target entity (Aspect Polarity Estimation). Hence, it is more challenging than the traditional sentence-level sentiment analysis where we predict the text's polarity as a whole. We treat this Aspect-Based Sentiment Analysis task as a combination of three different subtasks, namely, Aspect Term Extraction, Phrase Detection, and Aspect Polarity Estimation. Each step of the Aspect-Based Sentiment Analysis pipeline is explained in detail in the coming sub-sessions.

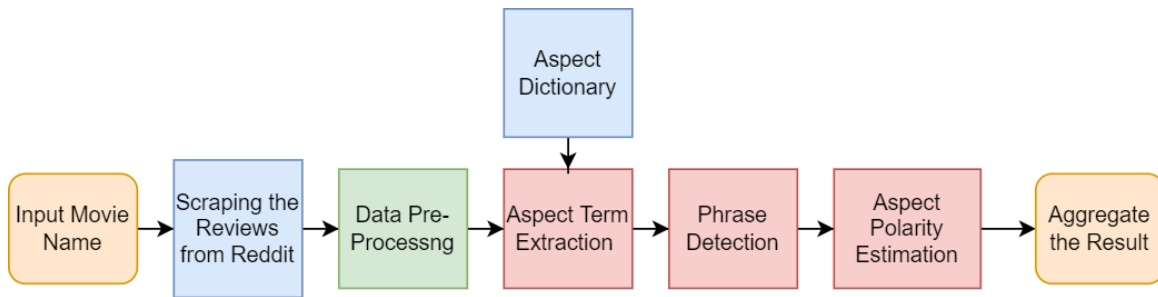


Figure 6. Aspect Based Sentiment Analysis Pipeline

Web Scraping

People's opinion about a movie is extracted in real-time from Reddit using the PRAW API [20]. As the name suggests PRAW is a python wrapper for Reddit API that enables us to scrape data from subreddit in real-time. PRAW does a lot of work for us. It lets us use a simple interface while it handles a lot of complex tasks in the background like rate limiting and organizing the JSON responses. To build a Reddit client and use PRAW API [20] we need to create a Reddit app and get the required authentication and details, for more information on the registration process check the appendix of this report. After getting the authentication details following steps are performed to get the public opinions about a given movie.

1. Setup Reddit Client using the authentication details generated during the registration process.
2. Go to the Sub-reddit hosting the official discussion for the movies.
3. Search for the Official Discussion post for the given movie.
4. If found Extract all the comments under that thread.

Data Pre-Processing

The comments scrapped may contain several entities which may not contribute to computing the sentiment of every aspect. Entities such as website URLs, hashtags, mentions, email IDs, and emojis are unwanted and can be removed. Following pre-processing steps are performed to remove unwanted components and reduce complexity.

1. Convert all the words to lower case in order to reduce the variations.
2. Remove all the Website URLs, emails, mentions, hashtags and emojis.
3. Remove all the punctuations.
4. Remove extra white spaces.

Stop words are not removed because for the Phrase extraction module it is necessary to maintain the grammar and lexical meaning of the review.

Aspect Term Extraction

To extract all the aspect terms from the review comments, we are using a pre-defined aspect category dictionary that consists of the main aspect categories and the aspect terms associated with each aspect category as shown in Table 3.

Noun phrases are extracted from the reviews as aspect terms might be mentioned using alternate names. For instance, a “movie” can be called a “film”. To extract such terms which are directly correlated with aspect terms we extract all nouns from the review comments. Thereafter, we find similarities between extracted nouns and pre-defined aspect terms to know the nouns that are alternate names used for aspect terms in the review comment. The steps to achieve this are mentioned below.

1. Use NLTK’s Part-of-speech (POS) tagging module [22] to assign part-of-speech tags to every token in the review. Part of speech might be nouns, pronouns, adjectives, adverbs etc.
2. Extract all the common nouns in the review.
3. Calculate Word Embeddings for the extracted nouns and the aspect terms in the dictionary using Spacy’s English Large Model [21].
4. Calculate Cosine Similarity between the extracted noun term’s Word Embeddings and the aspect term’s Word Embeddings.
5. If the similarity score of (aspect term, extracted noun) pair with the maximum score is greater than a pre-defined threshold value then the extracted noun term is considered as a synonym used to mention the corresponding aspect term.

In addition to different aspects of film people usually talk about the actors or other crew members by mentioning their names. We use NLTK’s Name Entity Recognition module [22] to identify all person names mentioned in the review and include that as an aspect term as well.

Aspect Category	Aspect Terms
Crew	act, actor, actress, acting, role, portray, acted, character, villain, performance, performed, casting, cast, casted, crew, hero, lead, writers, protagonist, performance
Direction	direction, directing, director, filming, filmmaking, filmmaker, cinematic, edition, cinematography, photography, frame, making, made, execution
Plot	storyline, story, tale, romance, moral, dialog, dialogues, end, storyteller, ending, starting, beginning, written, fantasy, storytelling, revenge, tone, climax, concept, moments, betrayal, plot, drama, dramatic, writing, twist, comedy, jokes, conclusion, message, transcripts, content,

	mystery, final sequence, first half, 1st half, 2nd half, second half, premise, fictional, historical, history, first act, second act, third act
Screenplay	scene, scenery, violence, screenplay, scenario, sets, set, action, stunt, shot, props, fight, visualization, costume, battle scenes, fighting scenes, camera, editing, edited, cuts, script, graphics, cgi, 3d, visual effect, sponces, visual, effects, special effect, animation, animated, narrative, setting, themes, pacing, final sequence
Music	lyric, sound, music, audio, musical, track, title track, sound effect, soundtrack, score, vocals

Table 3. Aspect Categories and Aspect Terms associated with each Aspect Categories

Phrase Extraction

Sentiment polarity for the extracted aspect terms is calculated on the phrases which are used to describe the aspects. Usually, adjectives and adverbs form such phrases that describe the nouns. To extract such describing phrases, we have used a Question-Answering model. We haven't trained a custom Question-Answering model from scratch but used a pre-trained model that's already available in the Hugging Face Library [7]. We used the DistilBERT-base-cased model that was trained on the SQuAD V1.1 dataset provided by Hugging Face Library [7] to perform Phrase Extraction.

The Question-Answering model requires two inputs. The question and the context from which the corresponding answer for the question is extracted. In our module we have used the pre-processed review comment as context and asked questions in the following format: *"how is {aspect_name}"*. This question extracts the describing phrases from the context for every aspect. For example, a review after pre-processing states "great acting dreadful editing". Using it as a context and asking questions such as "how is acting" and "how is editing" will yield answers like "great" and "dreadful" respectively.

Aspect Polarity Estimation

Sentiment Analysis is contextual mining of text which identifies and extracts subjective information in the text data. It results in one of the positive, negative, or neutral sentiments as shown in Figure 7. We have used a supervised approach to identify the sentiment of describing phrases that we extracted in the previous step for each aspect.

We finetuned BERTweet [23] model that is pre-trained on the Twitter dataset to create the Aspect Polarity Estimation module. The BERTweet [23] has the same architecture as the BERTbase model [4], which is trained with a masked language modelling objective. The corpus used to pre-train BERTweet [23] model consists of 850M English Tweets (16B word tokens) and this model is the first public large-scale model pre-trained for English Tweets.

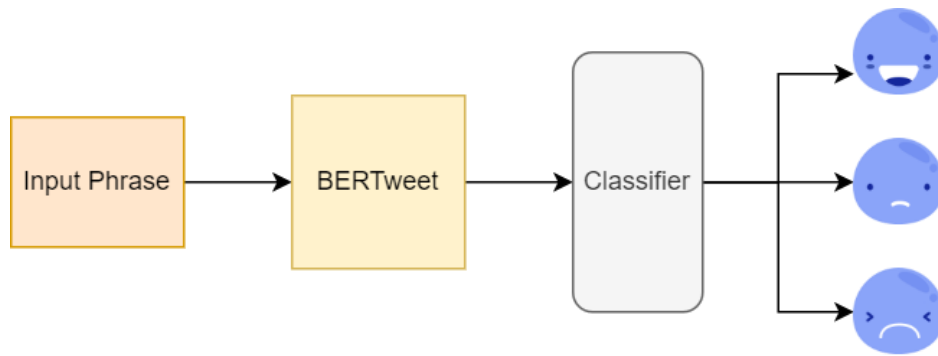


Figure 7. Aspect Polarity Estimation Network

We finetuned the BERTweet model using the upgraded version of moviesLarge dataset and the IMDb sentence polarity dataset [24]. The 8732 phrase, polarity pairs from the moviesLarge dataset and the 1000 sentence, polarity pairs from the IMDb sentence polarity dataset was split into train and validation data. 90 percent of the data was used to finetune the BERTweet model and the remaining 10 percent was used as validation data.

We used the Adam optimizer with a learning rate 10^{-7} to train the model. The model was finetuned for 25 epochs and used early stopping to prevent the model from getting overfitted. Figure 8 and Figure 9 illustrate the accuracy and loss curve obtained during model training respectively.

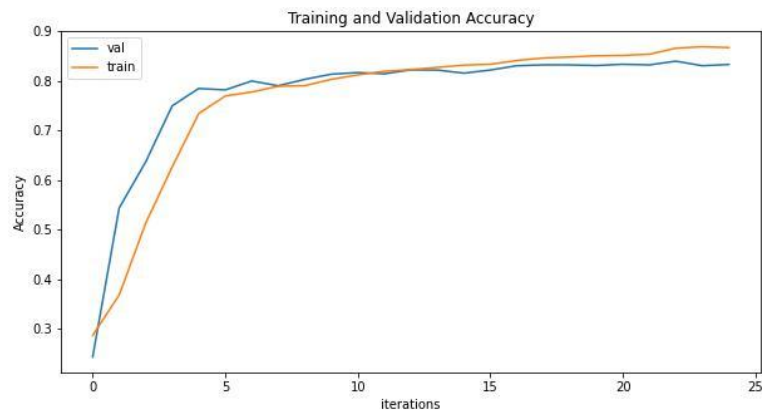


Figure 8. Training and Validation accuracy curve.

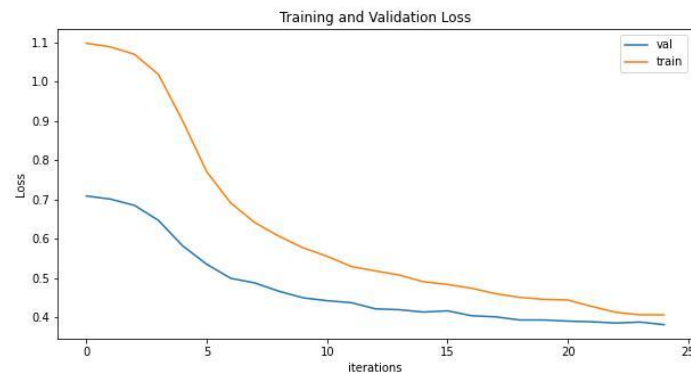


Figure 9. Training and Validation Loss curve

5.4 Evaluation Results

Aspect Term Extraction

We have evaluated our Aspect Term Extraction module using the 780 (aspect term, sentence) pairs on the upgraded version of the Movie20 dataset. Table 4 shows the Precision, Recall, F1 score, and accuracy of our Aspect Category Detection Module. We were able to achieve an overall accuracy of 85% on the Movie20 dataset.

Model	Precision	Recall	F1 Score	Accuracy
Aspect Term Extraction	0.727	0.828	0.775	0.849

Table 4. Aspect Category Detection Result

Aspect Polarity Estimation

We have evaluated our Aspect Polarity Estimation module using the 780 (phrase, polarity) pairs on the upgraded version of the Movie20 dataset. Table 5 shows the classification report for the Aspect Polarity Detection module. We were able to achieve an overall accuracy of 94% on the Movie20 dataset.

	Precision	Recall	F1 Score
Negative	0.96	0.92	0.94
Neutral	0.68	0.85	0.76
Positive	0.97	0.97	0.97
Accuracy			0.94
Macro Average	0.87	0.91	0.89
Weighted Average	0.95	0.94	0.94

Table 5. Classification Report of Aspect Polarity Detection Module

5.5 Deployment

For inference, the sentiment analysis pipeline will take the movie name as input and scrape all the reviews written from the official discussion post for the given movie name. Then with the help of GPU, the pipeline processes the review comments parallelly in batch and aggregates the fine-grained sentiment analysis results. After that, the aggregated result is used to plot the Aspect-Based Sentiment Analysis graph and pass that as an output to the chatbot.

5.6 Limitation

Currently, the inference speed is limited by the speed of Scraping API provided by Reddit. In addition to that, the sentiment analysis module requires the help of GPU to process the thousands of review comments real-time.

6 NLP Recommender system

MovieBot is also equipped with an API to call the NLP based recommender system, as MovieBot does not have a database of users to track an individual's ratings for a movie. The user can describe the aspects of the film to the chatbot which will extract the necessary slots from the conversation and use those to search against the database of movies to find the movies which best match the inputs provided.

This NLP based approach has become much less common as it can be quite naive in its search and is unable to take into account the types of movies the user has liked in the past or even what movies they have seen before. In order to compare the similarity of movies to each other and descriptions from users, we need to create embedding vectors which can be used to calculate cosine similarity. There are many methods to do this and several were explored over the course of the project.

The most basic form of embedding is Bag of Words, which uses a combination of the words in a sentence or in this case the fields of the movie to create the embedding which describes the title. This embedding can then be vectorized in many different ways including Count Vectorization, Tf-Idf (Term Frequency - Inverse Document frequency) or using more complex transformers to create embeddings like BERT.

6.1 Dataset

A critical piece of any recommender system is the dataset underlying it. There are many movie datasets available and finding the right one for the use case is an important step. The two largest datasets which were freely available were the IMDb [25] and MovieLens [26] datasets. For the initial experiments with different recommendation methods were conducted using a small 250 movie dataset pulled from IMDb. This dataset contained a wide variety of information about the movie including: genre, directors, actors, writers, a short summary of the plot, release year and the production company for the film; to name a few.

The decision was made early on in the project for the MVP a reduced set of categories should be used. Genre, director and actors were chosen as the best fields to begin with as these are the most commonly referenced parts of a movies production team. Plot was also considered but in researching and discussing with the team about the development of the NLU it was found that plot is a very difficult type of slot to detect for a task oriented chatbot design as it is much harder to define the bounds of and prepare a dataset for training a neural network to accurately learn to correctly identify what the user is describing. As such the decision was made to drop plot as a potential field for the recommendation search. When reviewing possible larger datasets it was found that many of the fields in the original smaller dataset were not readily available in the larger datasets. The IMDb datasets also lacked the actors column in its files, as such the dataset while having the advantage of being updated daily was unsuitable for use in this implementation.

MovieLens also has multiple datasets designed for different tasks, the required fields were found spread across two different datasets. The MovieLens 25M Dataset has 25 million reviews for 62,000 movies, this dataset contains a file with the movie titles and genres. The MovieLens Tag Genome Dataset 2021 which contains 10.5 million computed tag-movie relevance scores also contains a JSON file with over 84,000 movies including title, director and actors for each. The data from these files was merged using

the movie title to generate the recommendation dataset. This new dataset did require some cleaning as not all movies had information in all of the fields. The data manually annotated for some more landmark movies in the dataset such as some recent films from the Marvel franchise, the remaining movies were scrubbed from the dataset. Following this the dataset contains over 53 thousand movies, with all the necessary fields for the recommendation system.

Title	Genre	Director	Actors
0 Toy Story (1995)	Adventure Animation Children Comedy Fantasy	John Lasseter	Tim Allen, Tom Hanks, Don Rickles, Jim Varney, John Ratzenberger, Wallace Shawn, Laurie Metcalf, John Morris, R.
1 Jumanji (1995)	Adventure Children Fantasy	Joe Johnston	Jonathan Hyde, Bradley Pierce, Robin Williams, Kirsten Dunst
2 Grumpier Old Men (1995)	Comedy Romance	Howard Deutch	Jack Lemmon, Walter Matthau, Ann-Margret, Sophia Loren
3 Waiting to Exhale (1995)	Comedy Drama Romance	Forest Whitaker	Angela Bassett, Loretta Devine, Whitney Houston, Lela Rochon
4 Father of the Bride Part II (1995)	Comedy	Charles Shyer	Steve Martin, Martin Short, Diane Keaton, Kimberly Williams, George Newbern, Kieran Culkin
5 Heat (1995)	Action Crime Thriller	Michael Mann	Robert De Niro, Al Pacino, Val Kilmer, Jon Voight, Tom Sizemore, Ashley Judd, Diane Venora, Natalie Portman

Figure 10. Sample values of the new dataset

6.2 Experimentation

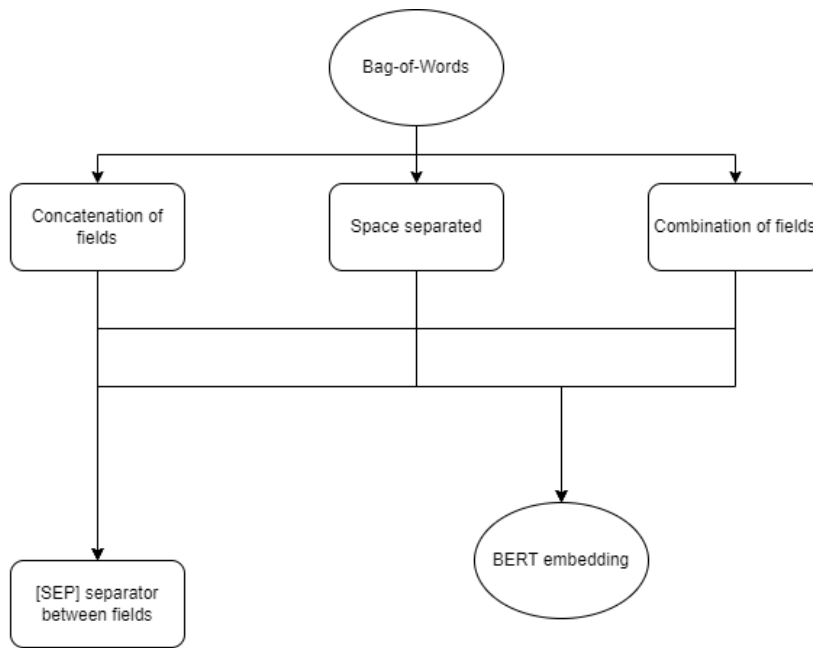
Over the course of the project multiple configurations of word and sentence embeddings were investigated. Based on the original reference [27] the words in each field were preprocessed to lowercase all words and remove any punctuation. Then all the words were concatenated to create unique words representing each field, these individual fields were then combined to create a “sentence” to represent the movie. This method was effective for the original purpose of the example when the search was using existing titles in the dataset to search for other similar titles to recommend however for our application this didn’t yield good results. The chatbot will result in a more relaxed and conversational style of speaking as a result the users: may not provide all fields; may not provide all information in a field or may not use the full names for actors and directors. This means that the word embeddings from the user are unlikely to match those of our dataset.

As a result of this a new configuration was used which did not concatenate all the words in a field together. Instead, the words were simply combined with spaces to separate each word and combined into a “sentence”. This method was able to better handle the user providing only some of the genres for the movie and referring to directors or actors by their first or last name only.

However, there were still instances where the system would match names outside the correct fields, specifically when actors were also the director for movies. To address this a new configuration was investigated which created an embedding for each of the possible combinations of the fields. E.g. genre and direct, genre and actors, director and actor etc. This method yielded the best results so far and helped to avoid the issue or repeat names in the majority of cases.

During development, the use of more modern methods such as GloVe embeddings and transformers were also investigated. GloVe embeddings were unable to handle the presence of the actors’ and directors’ names in the data, with the limited time available the decision was made not to attempt to train GloVe embeddings specifically for this project. Instead the investigation moved on to using the BERT transformer to generate embeddings for the movies.

To start the same “sentence” structure as used for the Bag-of-Words implementation, over the course of development many different configurations were tested, such as leaving the words as individual tokens, and creating embeddings for the different combinations for categories. It was also investigated if concatenating the embeddings of the individual fields to create the different combinations would yield different results. Throughout these experiments using the “[SEP]” token between different categories was also attempted to see if it influences the quality of the results.



Genre	director	actors	Bag-of-Words
crimedrama	stevenspielber	benstiller	crimedramastevenspielbergbenstiller
actionthriller	georgelucas	jamesmcavoy	actionthrillergeorgelucasjamesmcavoy

Genre	director	actors	Bag-of-Words
crime drama	steven spielberg	ben stiller	crime drama steven spielberg ben stiller
action thriller	george lucas	james mcavoy	action thriller george lucas james mcavoy

Genre	director	actors	Bag-of-Words	BoW_genre_director	BoW_genre_actors
crime drama	steven spielberg	ben stiller	crime drama steven spielberg ben stiller	crime drama steven spielberg	crime drama ben stiller
action thriller	george lucas	james mcavoy	action thriller george lucas james mcavoy	action thriller george lucas	action thriller james mcavoy

Genre	director	actors	Bag-of-Words
crime drama	steven spielberg	ben stiller	crime drama [SEP] steven spielberg [SEP] ben stiller
action thriller	george lucas	james mcavoy	action thriller [SEP] george lucas [SEP] james mcavoy

Figure 11. Embedding experimentation with data examples

6.3 Evaluation

Evaluation of recommender systems is a difficult task; recommendations are generally more of a qualitative measurement than a quantitative one. It is difficult to know how well a recommender will perform without actually exposing it to users and gathering feedback. There are some metrics which can be used for systems like collaborative filtering and content-based filtering systems because they have a database of users to draw from to simulate recommendation scenarios by masking random movies which we know the user has watched in the past. With our recommendation system we don't have this data, the system is quite a basic and naive method of recommendation which doesn't know the user's history and only aims to match the search terms as closely as possible.

Several methods were attempted to provide some sort of quantification of the recommendation's accuracy. The first was based on those used for the collaborative filtering example, here random parts of a movies embedding "sentence" were removed and run through the recommendation system. Top 10 results were checked to ensure that the original movie title still appeared in the list even after some information was removed from the embedding. This allows us to confirm that the embedding is robust to missing information in the search terms from users.

We also conducted some more qualitative test by using a python function which searched the dataset for RegEx matches in the dataset and to ensure the search was returning all of the possible search results. This was a manual task, as a result only a small subset of searches were conducted and compared. This along with having users from outside the project development team allowed us to identify issues with the different embedding methods and get a general sense of the quality of the experience for the users.

Following the experiments and evaluation it was found that BERT embeddings didn't perform well when parts of the "sentence" were omitted from the input. As a result, BERT embeddings didn't meet the requirements for the implementation. This does make some sense as there is little meaning shared between the fields as they are simply names and genres, there is very little context information to be drawn from the sentences.

As a result of these finding the classical method using Bag-of-Words was implemented for the final version of the system, using Tf-Idf vectorization as it was found this produced the best overall experience for users and reduced the issues caused by common names in search terms and movies with less tokens in the "sentences" scoring higher in similarity than those with a greater number of matches with more tokens in the "sentences".

6.4 Deployment

The final implementation of the system uses the classical Bag-of-Words method to create the embeddings for the titles with the added improvement of creating a column for each possible combination of categories. Depending on the input of the user into the recommendation API the required column is selected and converted into a vector using Tf-Idf vectorization. It was found in testing that Tf-Idf avoided issues with common names matching more often and prioritizing shorter "sentences" over those that matched more exactly. The search term is also run through the Tf-Idf vectorization and compared against all the movies in the database using cosine similarity. The top 5 highest scoring

indexes are selected, the information about these movies are then pulled from the database and used to create formatted strings which are passed in a list to the UI to be displayed to the user.

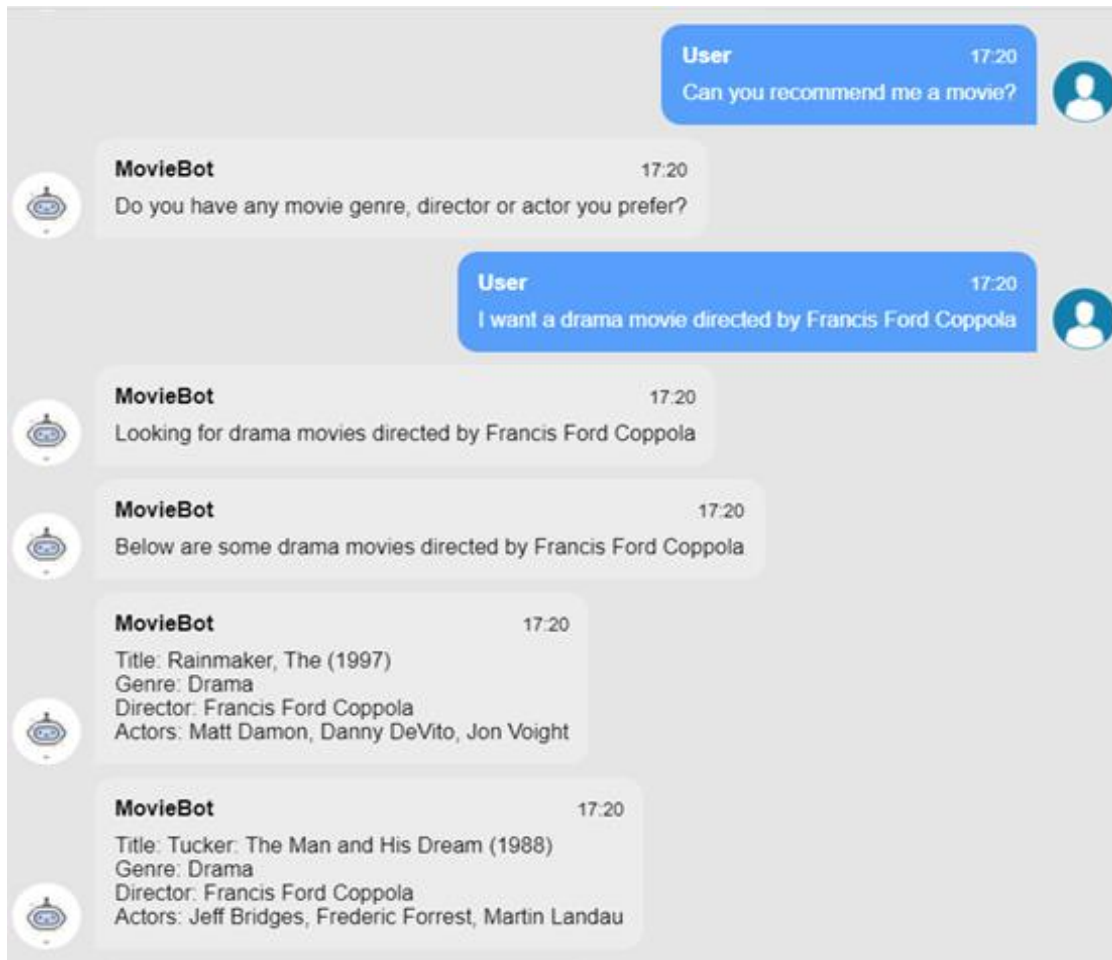


Figure 12. Sample of a conversation using the recommendation API

6.5 Limitations

There are some limitations with the current system, the dataset is only up until 2019 so some of the more recent movies in the last few years and therefore new actors and directors are not represented in the dataset.

Additionally, while the current method of embeddings is able to provide good results for most search terms there are cases where it struggles to give the best results. When a user provides an actors and director for the search terms, if one of these people has both acted and directed in a single film this but does not have the other person, this movie can still be recommended because there are twice the number of matched tokens. This shows that the embedding are not able to understand the importance of order in the search terms, and is simply matching their existence. The same can happen if the actor has a common names or siblings in the industry. E.g. Ben Affleck can return results where Casey Affleck acted with Ben Stiller for example.

7 User Interface & Integration

The chatbot user interface is being deployed using the Flask app. The Flask app exposes a REST API for the front-end application to send or receive information. It hosts most of the components within the architecture and interacts with them, directing data flow among the components. The front-end was developed using HTML and CSS. Figure 13 illustrates the User Interface design of our chatbot.

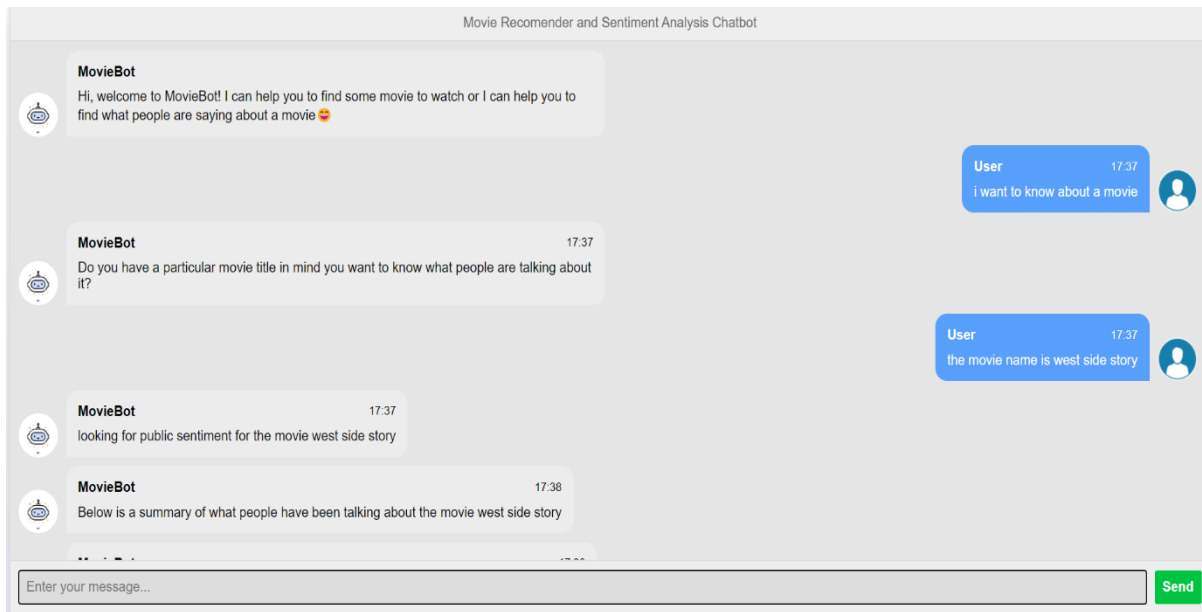


Figure 13. Chatbot User Interface

The chatbot, Aspect-Based Sentiment Analysis and Recommender modules are built on the python backend. The Aspect-Based Sentiment analysis and recommender module work independently in the backend. Whenever a user utterance is being captured in the chatbot user interface it will be processed by the chatbot backend module to extract intents and slots. The conversational states will be updated, and corresponding API modules will be called. With the updated states and API results, the NLG module then generates a chatbot response. Finally, the chatbot response will be sent back to the chatbot user Interface for interaction. This UI is able to provide a simple and effective interface for users to converse with our chatbot and perform the relevant tasks.

8 Conclusion

Given the limited time frame for us to work on this project, our team has successfully developed a chatbot system that is able to understand user's utterances correctly and to interact with the user in a fluent and natural way. Upon users request the chatbot is able to identify the movie of interest to the user and provide informative insights into the audience's sentiments about the movie. On the other hand, the chatbot is able to handles requests for movie recommendation and identify the categories of interest to generate a relevant list of recommendation for the user.

Within the task-oriented chatbot architecture we have successfully fine-tuned a BERT model for joint intent and slots classifications. We have pre-trained and further fine-tuned a T5 model that is capable of controlled text generations for the NLG module. We have also developed a rule-based dialogue management system that manages the interaction between different components of the chatbot system. We have designed a pipeline to extract important aspect terms from audience reviews and fine-tuned a BERT model to predict the polarity of the extracted aspect term. We have developed a method of word and sentence embeddings for the movie dataset which can be used to derive recommendations based on the user's described search terms.

In summary the chatbot meets the requirements set out in the MVP for the project, providing an baseline system which can be further developed into a market ready product for release in the media domain.

9 Future Improvements

Below are some future improvements that can be made to the components of the chatbot system:

Components	Description
Chatbot NLU & DST	Adopt a more advanced approach where the NLU and DST are treated as a single Question Answering (QA) task in detecting the intents and slots while keeping track of the conversation history at the same time. This may be useful for multi-intent and multi-domain chatbot applications as the NLU and DST tasks are performed concurrently.
Chatbot NLU & NLG	Improve the dynamicity of conversation by increase the dataset used for both the NLU and NLG. With more types of intent, slots and system actions recognizable by the NLU and NLG, the chatbot can deal with more complicated request or conversation with the user.
Aspect-Based Sentiment Analysis	Extract review comments from multiple social media platforms to produce more accurate analysis of public opinion.
Recommender System	Move to a live dataset so that we can make recommendations based on the latest movies. Develop a better system for embeddings which is able to handle more fields, understand the importance of order, and the concept of names.

10 References

- [1] Jiao Liu, Yanling Li and Min Lin, “Review of Intent Detection Methods in the Human-Machine Dialogue System” in Journal of Physics: Conference Series 2019
<https://iopscience.iop.org/article/10.1088/1742-6596/1267/1/012059>
- [2] H. Weld, X. Huang, S. Long, J. Poon, S. C. Han, “A survey of joint intent detection and slot-filling models in natural language understanding” in Computation and Language 2021
<https://arxiv.org/abs/2101.08091v3>
- [3] Qian Chen, Zhu Zhuo, Wen Wang, “BERT for Joint Intent Classification and Slot Filling” in Computation and Language 2019 <https://arxiv.org/abs/1902.10909v1>
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” in Computation and Language 2019
<https://arxiv.org/abs/1810.04805v2>
- [5] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, Joseph Dureau, SNIPS dataset, “Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces” in Computation and Language 2018 <https://github.com/sonos/nlu-benchmark>, <https://arxiv.org/abs/1805.10190v3>
- [6] Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. “An evaluation dataset for intent classification and out-of-scope prediction”. In Proceedings of EMNLP-IJCNLP, CLINC150 dataset <https://archive.ics.uci.edu/ml/datasets/CLINC150>
- [7] Hugging Face, <https://huggingface.co/>
- [8] Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, Dawei Song, “A Survey of Controllable Text Generation using Transformer-based Pre-trained Language Models” in Computation and Language 2022
<https://arxiv.org/abs/2201.05337v1>
- [9] Hao Zhou, Minlie Huang, Xiaoyan Zhu, “Context-aware Natural Language Generation for Spoken Dialogue Systems” COLING 2016 <https://aclanthology.org/C16-1191/>
- [10] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, Steve Young, “Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems” in Computation and Language 2015, <https://arxiv.org/abs/1508.01745v2>
- [11] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, Richard Socher, “CTRL: A Conditional Transformer Language Model for Controllable Generation” in Computation and Language 2019, <https://arxiv.org/abs/1909.05858v2>
- [12] Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, Jianfeng Gao, “Few-shot Natural Language Generation for Task-Oriented Dialog” in Computation and Language 2020,
<https://arxiv.org/abs/2002.12328v1>
- [13] Mihir Kale, Abhinav Rastogi, “Text-to-Text Pre-Training for Data-to-Text Tasks” in Computation and Language 2021, <https://arxiv.org/abs/2005.10433v3>

- [14] Mihir Kale, Abhinav Rastogi, "Template Guided Text Generation for Task-Oriented Dialogue" in Computation and Language 2020, <https://arxiv.org/abs/2004.15006v2>
- [15] Data-to-Text Generation Model Ranking based on MultiWoz 2.1 dataset <https://paperswithcode.com/sota/data-to-text-generation-on-multiwoz-2-1>
- [16] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" in Machine Learning, Computation and Language 2020 <https://arxiv.org/abs/1910.10683v3>
- [17] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, Pranav Khaitan, "Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset" in Computation and Language 2020, <https://arxiv.org/abs/1909.05855v2>
- [18] Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, Dilek Hakkani-Tur, "MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines" in Computation and Language 2019, <https://arxiv.org/abs/1907.01669v4>
- [19] Vaibhav Bajaj, Kartikey Pant, Ishan Upadhyay, Srinath Nair, Radhika Mamidi, "TEASER: Towards Efficient Aspect-based SEntiment Analysis and Recognition", RANLP 2021, <https://aclanthology.org/2021.ranlp-1.13/>
- [20] PRAW: The Python Reddit API Wrapper, <https://praw.readthedocs.io/en/stable/>
- [21] spaCy en_core_web_lg, <https://spacy.io/models/en>
- [22] Natural Language Toolkit, NLTK, <https://www.nltk.org/>
- [23] Dat Quoc Nguyen, Thanh Vu, Anh Tuan Nguyen, "BERTweet: A pre-trained language model for English Tweets", EMNLP 2020, <https://aclanthology.org/2020.emnlp-demos.2/>
- [24] Sentiment Labelled Sentences Data Set, <https://www.kaggle.com/datasets/marklv/sentiment-labelled-sentences-data-set>
- [25] IMDb, "IMDb Datasets", [Online]. Available: <https://www.imdb.com/interfaces/>
- [26] GroupLens, "MovieLens," [Online]. Available: <https://grouplens.org/datasets/movielens/>
- [27] J. Ng, "Content-based Recommender Using Natural Language Processing (NLP)," [Online]. Available: <https://www.kdnuggets.com/2019/11/content-based-recommender-using-natural-language-processing-nlp.html>.

11 Appendix A: Handcrafted Inform Intents Dataset

The 3 types of inform intents under the NLU handcrafted datasets are:

- INFORM_INTENT MOVIE_SENTIMENT – indicates movie sentiment API domain
- INFORM_INTENT MOVIE_RECOMMENDER – indicates movie recommendations API domain
- INFORM – user provide slot values

The annotated user utterance samples are created to cover the following combinations of slot values detectable:

Intent	Slots Detected in Sample
INFORM_INTENT MOVIE_SENTIMENT	-
INFORM_INTENT MOVIE_SENTIMENT	title
INFORM_INTENT MOVIE_RECOMMENDER	-
INFORM_INTENT MOVIE_RECOMMENDER	genre
INFORM_INTENT MOVIE_RECOMMENDER	director
INFORM_INTENT MOVIE_RECOMMENDER	actor
INFORM_INTENT MOVIE_RECOMMENDER	genre, director
INFORM_INTENT MOVIE_RECOMMENDER	genre, actor
INFORM_INTENT MOVIE_RECOMMENDER	director, actor
INFORM_INTENT MOVIE_RECOMMENDER	genre, director, actor
INFORM (SENTIMENT)	title
INFORM (RECOMMENDER)	genre
INFORM (RECOMMENDER)	director
INFORM (RECOMMENDER)	actor
INFORM (RECOMMENDER)	genre, director
INFORM (RECOMMENDER)	genre, actor
INFORM (RECOMMENDER)	director, actor
INFORM (RECOMMENDER)	genre, director, actor

As the dataset is created manually, we try to include as many variations as possible of user expression for each of the scenario listed above.

12 Appendix B: Handcrafted Control Code Dataset for Text Generation

The control codes are predefined as follows. To generate the dataset, the slot values (“var1”, “var2”, “var3”) are randomly replaced with actual movie titles, genre, director and actor names.

System Actions	Predefined Control Code
REQUEST	MOVIE_RECOMMEND REQUEST (genre = ? ; directed_by = ? ; starring = ?) MOVIE_SENTIMENT REQUEST (title = ?)
NOTIFY_SEARCH	MOVIE_RECOMMEND NOTIFY_SEARCHING (genre = var1) MOVIE_RECOMMEND NOTIFY_SEARCHING (director = var1) MOVIE_RECOMMEND NOTIFY_SEARCHING (actor = var1) MOVIE_RECOMMEND NOTIFY_SEARCHING (genre = var1 ; director = var2) MOVIE_RECOMMEND NOTIFY_SEARCHING (genre = var1 ; actor = var2) MOVIE_RECOMMEND NOTIFY_SEARCHING (director = var1 ; actor = var2) MOVIE_RECOMMEND NOTIFY_SEARCHING (genre = var1 ; director = var2 ; actor = var3) MOVIE_SENTIMENT NOTIFY_SEARCHING (title = var1)
NOTIFY_SUCCESS	MOVIE_RECOMMEND NOTIFY_SUCCESS (genre = var1) MOVIE_RECOMMEND NOTIFY_SUCCESS (director = var1) MOVIE_RECOMMEND NOTIFY_SUCCESS (actor = var1) MOVIE_RECOMMEND NOTIFY_SUCCESS (genre = var1 ; director = var2) MOVIE_RECOMMEND NOTIFY_SUCCESS (genre = var1 ; actor = var2) MOVIE_RECOMMEND NOTIFY_SUCCESS (director = var1 ; actor = var2) MOVIE_RECOMMEND NOTIFY_SUCCESS (genre = var1 ; director = var2 ; actor = var3) MOVIE_SENTIMENT NOTIFY_SUCCESS (title = var1)
NOTIFY_FAILURE	MOVIE_RECOMMEND NOTIFY_FAILURE (genre = var1) MOVIE_RECOMMEND NOTIFY_FAILURE (director = var1) MOVIE_RECOMMEND NOTIFY_FAILURE (actor = var1) MOVIE_RECOMMEND NOTIFY_FAILURE (genre = var1 ; director = var2) MOVIE_RECOMMEND NOTIFY_FAILURE (genre = var1 ; actor = var2) MOVIE_RECOMMEND NOTIFY_FAILURE (director = var1 ; actor = var2) MOVIE_RECOMMEND NOTIFY_FAILURE (genre = var1 ; director = var2 ; actor = var3) MOVIE_SENTIMENT NOTIFY_FAILURE (title = var1)
REQ_MORE	MOVIE_RECOMMEND REQ_MORE MOVIE_SENTIMENT REQ_MORE
GOODBYE	MOVIE_RECOMMEND GOODBYE MOVIE_SENTIMENT GOODBYE

As the dataset is created manually, we try to include as many variations as possible of system utterances for each of the scenario listed above.

13 Appendix C: BERT for Joint Intent and Slot Classification Fine-Tuning Setup

Setup	Parameters/Value
Dataset	Handcrafted + Basic Intents from CLINC150
Dataset Size	790 user utterances (6,754 token BIO Tags)
Train-Validation-Test Split	70%-15%-15% split stratified on intent types
Input Token Max Length	32
Batch Size	128
Epochs	20
Optimizer	Adam with constant LR 5e-5
Model Callback	Lowest Validation Total Loss

14 Appendix D: T5-small model Pre-Training and Fine-Tuning setup for text generation

Setup	Pre-Training	Fine-Tuning
Dataset	SGD Dataset	Handcrafted Dataset
Dataset Size	231,642 system utterances	1,030 system utterances
Train-Validation Split	70%-30% split	70%-30% split
Input Token Max Length	128	64
Batch Size	48	64
Epochs	10	10
Optimizer	Adam with constant LR 5e-4	Adam with constant LR 5e-4
Model Callback	Greedy Search BLEU	Greedy Search BLEU Beam Search BLEU Top-p (nucleus) Sampling BLEU

15 Appendix E: Reddit App Creation Steps

1. Go to [App Preferences](#) and login to your reddit account using your User id and Password.
2. Click on create another app button located at bottom left of the page.
3. Fill out the required details, make sure to select script and – click create app as shown in the Figure below.

create application

Please [read the API usage guidelines](#) before creating your application. After creating, you will be required to [register](#) for production API use.

name


☐ web app A web based application
☐ installed app An app intended for installation, such as on a mobile phone
☒ script Script for personal use. Will only have access to the developers accounts

description

about url

redirect uri

4. Make a note on the **personal use script** and **secret** tokens shown below. This is later passed as credentials in the python script while creating the Reddit Client.



MyBot
 personal use script
WXLU6PBXkGZOAw

Personal use for pulling rec

[change icon](#)

secret

name

description

about url

redirect uri

[delete app](#)

16 Appendix F: Installation Instructions

1. To Run our chatbot you need a system with GPU and CUDA installed. To install CUDA follow the instruction in [this link](#).
2. Download and install [Anaconda](#).
3. Create a new anaconda environment using.
conda create --name movie_bot
4. Activate the conda environment using.
conda activate movie_bot
5. Install pytorch with GPU support using.
conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch
6. Install required python packages using the command.
**cd path\to\chatbot_folder\Chatbot\
pip install -r requirements.txt**
7. Install Spacy English Large model using.
python -m spacy download en_core_web_lg
8. Download and extract models from [this link](#).
9. Copy the two folders (nlu_nlg_models, Sentiment_Analysis) inside the zip file to
path\to\chatbot_folder\Chatbot\Models
10. Run the Chatbot app using.
python app.py
11. Copy pastes the link ("http://localhost:8000/") displayed in the terminal to a web browser.